

PATENT

Docket No. RSW920030089US1

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

INVENTORS:	David A. Selby	Confirmation No. 1932
APPLICATION NO.	10/806,204	
FILED:	March 22, 2004	Examiner: A. Boyce
CASE NO.	RSW9200130089US1	Group Art Unit: 3623

TITLE: **SYSTEM, METHOD, AND COMPUTER PROGRAM
PRODUCT FOR INCREASING THE EFFECTIVENESS OF
CUSTOMER CONTACT STRATEGIES**

FILED ELECTRONICALLY

MAIL STOP AMENDMENT
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

DECLARATION OF DAVID A. SELBY UNDER 37 C.F.R. §1.131

Sir:

I, David A. Selby, hereby declare as follows:

1. I am the inventor of the subject matter claimed in the above-identified patent application.
2. I have been informed that U.S. Patent Application Publication No. 2005/0010472 to Quatse et al. ("Quatse") has been cited as prior art by the United States Patent and Trademark Office with respect to the above-identified patent application. I also have been informed that the filing date of Quatse is July 8, 2003.
3. This declaration is to establish conception and reduction to practice of the invention claimed in claims 1-24 of the above-identified application in the United States prior to July 8, 2003.

4. Prior to July 8, 2003, I conceived of and developed (i.e., reduced to practice) a working software program that met all of the limitations of claims 1-24 of my patent application, as more particularly detailed below.

5. Exhibit A attached hereto is an Invention Disclosure, prepared by me concerning the development of the invention disclosed and claimed in the above-identified patent application. The date appearing on the original of this document has been redacted from the copy attached hereto. However, the date is prior to July 8, 2003 and accurately reflects a date on which the document existed.

6. Exhibit B attached hereto is a print-out of computer program code for the computer program that is the basis of the present invention ("Vanquish.cpp"). Work on this program began on December 4, 2001 and was completed on or before December 31, 2002, which is prior to the July 8, 2003 filing date of Quatse.

7. The correlation between the steps of independent claim 1 and the code in Exhibit B is as follows:

- "performing a first sort on all unselected events to form a pending event list, so that the events are ordered sequentially by their values, with the highest valued event being at the top of the pending event list" -- data is pre-sorted in RDBMS engine and pulled into the computation space in the subroutine calls lines 264-295.
- "selecting the highest valued unselected event upon the occurrence of a predetermined trigger" -- this is performed in the subroutine specified in line 356 getnextval
- "recomputing the values of each event after the selection of the highest valued unselected event" -- this is performed in lines 419-436

- “moving the highest valued unselected event, after performance of the recomputing step, to the top of the pending event list without performing a second sort of the entire pending event list” -- this is performed in the subroutine line 356 getnextval by lazy evaluation and is completed in line 447-484 incnextval subroutine

8. I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment or both, under Section 1001 of Title 18 of the United States Code, and that such willful statements may jeopardize the validity of the application or any patent issued thereon.

17th July 2009
Dated

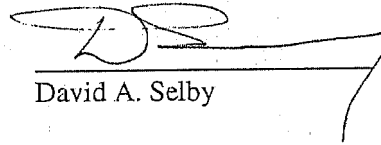

David A. Selby

EXHIBIT A



Disclosure RSW8-2003-0180

Prepared for and/or by an IBM Attorney - IBM Confidential

Created By David Selby On [REDACTED] 10:34:18 AM CET

Last Modified By David Selby On [REDACTED] 06:12:17 AM MST

Required fields are marked with the asterisk (*) and must be filled in to complete the form .

*Title of disclosure (in English)

A method for optimisation of marketing resources

Summary

Status	Under Evaluation
Final Deadline	
Final Deadline	
Reason	
*Processing Location	Raleigh - RSW
*Functional Area	<input type="text" value="select"/> (GS-Mike Haydoch/BICNS) GS-Mike Haydoch/BICNS
Attorney/Patent Professional	Gregory Doudnikoff/Raleigh/IBM
IDT Team	<input type="text" value="select"/> Joel R Grosh/San Diego/IBM
Submitted Date	[REDACTED] 10:49:24 AM CET
*Owning Division	<input type="text" value="select"/> GS
*Line of Business	BCS - Business Consulting Services Primary Inventor's Line of Business (LoB)
*Industry/Sector	Dist - Retail
*Competency	Consulting
Incentive Program	
Lab	
*Technology Code	687C
PVT Score	

Inventors with a Blue Pages entry

Inventors: David Selby/UK/IBM

Inventor Name	Inventor Serial	Div/Dept	Inventor Phone	Manager Name
Selby, David A (David)	086349	70/703211	7-246723	Pressley, Carl Nicholas

> denotes primary contact

Inventors without a Blue Pages entry

IDT Selection

Attorney/Patent Professional Gregory Doudnikoff/Raleigh/IBM
IDT Team Joel R Grosh/San Diego/IBM
Response Due to IP&L [REDACTED]

REDACTED

REDACTED

REDACTED

REDACTED

To: David Selby/UK/IBM@IBMGB
cc: Carl Nicholas Pressley/UK/IBM@IBMGB
Subject: Submission of Disclosure Number RSW820030180
Security: ☒ IBM Confidential
Importance: ☐ Urgent ☐ Normal

DISCLOSURE NUMBER: RSW820030180
TITLE: A method for optimisation of marketing resources

REDACTED

To: Joel R Grosh/San Diego/IBM
cc: Gregory Doudnikoff/Raleigh/IBM
Subject: Evaluation of Invention Disclosure RSW820030180
Security: ☒ IBM Confidential
Importance: ☐ Urgent ☐ Normal

DISCLOSURE NUMBER: RSW820030180
TITLE: A method for optimisation of marketing resources

REDACTED

Disclosure: A method for optimisation of marketing resources

Introduction

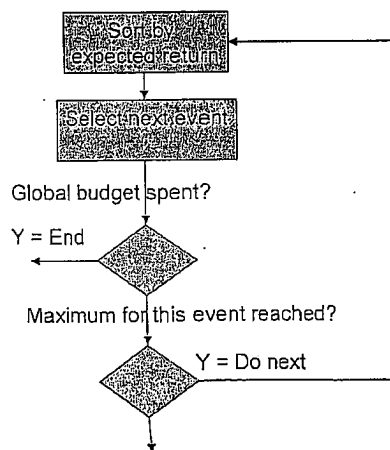
This disclosure describes a method for solving programmatically the business issue of optimising marketing resources. That is given a set of constraints of marketing materials, customers, budgets etc. An optimal stream of communication is selected which affords the "best" return on investment over time. This over approach has been disclosed in Docket Number P23,426 USA **SYSTEM AND METHOD FOR INCREASING THE EFFECTIVENESS OF CUSTOMER CONTACT STRATEGIES**

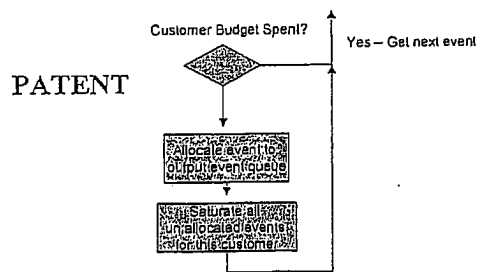
99-079
=
Bieblin

This disclosure makes further claims, by describing a computer-based method, which optimal solves the problem described in the above, by utilising a novel extension to a well-known operational technique called a greedy heuristic. (See www.nist.com for formal definitions)

The method

The objective of the process is to develop a per customer contact stream, that is from the total list of marketing events available to an individual we wish to select a subset which maximises profit for the minimal cost, taking into account a set of business constraints. We propose this can be achieved in the following way:





This approach adopts a well know technique, a greedy heuristic. Although, due to the circumstances involved in the data it is possible to omit the intermediate sort between each application of the saturation. Saturation in nearly all applications reduces the target scores; the knowledge of this allows us to make a simple comparison of the next value to be selected with the subsequent. If this test fails then a sort is required, if this test passes then no sort is required. Omitting this continual sort significantly increases the performance of the technique and allows near real-time application of the approach.

Our Claims

We Claim an approach which:

1. Will solve efficiently enough to enable the solution to be applied to each individual, and therefore improve over the micro groups discussed in the referenced application.
2. By sorting only when required, the algorithm becomes highly scalable.
3. The algorithm achieves a solution that alleviates the need for linear programming.
4. The algorithm can be applied in a real-time application, to recompute the best contact strategy each time the underlying data is changed by customer interaction.

1. Due to the way the problem was being mathematically solved using a linear program in Bibelnieks it was necessary to solve for subsets of customers only on an irregular basis. As it would take large amounts of batch time to solve for a reasonably large portfolio of customers.

- a. The other implication of the Bibelnieks method was that you budget on a per Micro asset group basis. Which can be construed to be a very unrobust approach.

The improvement I have been able to make is to using a greedy Heuristic approach I have been able to solve for each individual customer, and therefore I can budget to an individual customer. Although I would argue the approach I use deviates from the written work enough to be unique, see (2)

2. The GH on its own would not necessarily give you a sufficiently large enough of a kick to allow to solve for an individual, but on examining the typical data used for this type of problem I have discovered a further optimisation.
3. I think Bibelnieks, please verify in case I missed talks about solving the problem once and using for a longish time. My approach resolves the problem once per day for a constrained time window, say 30-90 days. So it is continually upto date. If you consider before Bibelnieks you would (and most still do) be in a six weeks cycle from generating a list of candidates and the mail dropping. Let alone planning customers, which I think by at least implication would be a long term planning activity.
4. The reason it is desirable to solve it once per day is that you need to be able to dynamically change the customers contact stream as soon as possible. Say it panned out the stream ended up hitting you with campaigns that sold you one particular item, say a winter coat. Well if I do it the Bibelnieks way and run my plan for the next few months, then one week in you buy a winter coat, every campaign you get after that point will be a waste of my marketing budget. Worse it will probably lead you to believe that I'm also stupid because you know we both know that you have a winter coat. So if you can update the plan the day something happens it becomes much more relevant for all subsequent contacts (campaigns). And improves the quality of the ROI from the stream of campaigns.

The GH flow is as follows:

From Bibelnieks you can see an individuals record would consist of a vector of scores for their propensity to respond, with some for of canalization and saturation applied, by a set of matrices. (See Bibelnieks figs 7-11) So you get, each being scored vertically. Normally a set of these would be input to the linear program plus the constraints.

Cannibalization and Saturation are taught by Bibelnieks in Figures 6-11 we would use the same or similar methodology in this approach. Now we deviate, Bibelnieks groups these customers into Micro segments based on some financial return curve for the group. We continue to work at an individual level, and also we operate a per customer budget based on the historical returns, are the ratio of marketing spend vs the amount of net profit for that customer.

Customer No	C1	C2	C3	C4	C5	C6	C7
123456	0.2	0.3	0.565	0.9	0.32	0.53	0.56
123457	0.1	0.8	0.63	0.5	0.21	0.52	0.7

(a) Create a pivot table from the above record, which looks like this:

The important thing to note is the chronological order of the campaigns has not been destroyed. Campaign 1 occurs at T+0 and Campaign 2 occurs at T+1 and Campaign 3 occurs at T+2 etc etc.

Customer No	Campaign	Expected Gain
123456	1	0.2
123456	2	0.3
123456	3	0.565
123456	4	0.9
123456	5	0.32
123456	6	0.53
123456	7	0.56
123457	1	0.1
123457	2	0.8
123457	3	0.63
123457	4	0.5
123457	5	0.21
123457	6	0.52
123457	7	0.7

(b) Sort by expected Gain. (The implication of this step is that you are optimizing the list of contacts globally. (ie: Independent of any one individual. Any event that creates the best expected gain will be top of the list)

Here a Radix sort is applied, The result is as follows:

Customer No	Campaign	Expected Gain
123456	4	0.9
123457	2	0.8
123457	7	0.7
123457	3	0.63
123456	3	0.565
123456	7	0.56
123456	6	0.53
123457	6	0.52
123457	4	0.5
123456	5	0.32
123456	2	0.3
123457	5	0.21
123456	1	0.2
123457	1	0.1

(fig 2)

Campaign	C1	C2	C3	C4	C5	C6	C7
Min Piece	1000.00	500.00	1000.00	1000.00	500.00	400.00	800.00
Max Piece	5000.00	1000.00	2000.00	3000.00	1500.00	1200.00	1500.00
Unit Price	0.62	0.56	0.43	1.03	0.93	1.20	1.04
Max Budget	3100.00	560.00	860.00	3090.00	1395.00	1440.00	1560.00
Min Budget	620.00	280.00	430.00	1030.00	465.00	480.00	832.00

(Apologies if this too small, its an

imbed so you can blow up) (fig 3)

We entered the optimization procedure with some budgets (or otherwise known as constraints) the table fig 3 above shows for this run the proposed constraints. Number of pieces are decided typically by print run cut offs, that is the printing trade price their work based on economic cut offs, which factor the size of the printing equipment, cost of setup etc. Basically this normally means the more you print the cheaper it gets. In a call centre operation, these cut off's exist but are predicated by the number of operators capable of making outbound email calls, or having the right skill to introduce that type of campaign. (You may have 100 operators but only 10 are specialized in selling mortgage products for instance) For each you have a minimum and a maximum (Upper and lower bound) The cost is the monetary amount it takes to execute each campaign, this yield a maximum budget Pieces times cost. If you add together all of these budget values, then you have the maximum permitted spend for this run of the application. Typically you would want to stay under this

value, the objective is to generate the most return by spending the least. Simple example the goal is to make 100 dollars by spending the least, clearly if I can spend 30 cents and make a 100 its better than spending 50 dollars and making 100. Sure I'm stating the obvious but it's the underlying fundamental in this process. The global budget is set to between (sum min budgets and sum max budgets) This is also helpful because in business the typical thing that happens is that times get tough so every department needs to take say a 10% cut, to meet targets. It's easy to just specify 10% less overall in this process.

(c) Now If the global budget met, we stop. As we allocate a campaign to an individual we sum the number the amount spent. This gives us a comparison number.

(d) We may stop if there nothing that will yield a predicted profit. In the column Expected gain, fig 2 (the arrow moves downward but points to the next entry to consider. If this top entry is negative as we work down the list and continually apply saturation, ie: Once we've done something to an individual subsequent events become less appealing, so we reduce the expected gains.

(e) if we've got nothing left in the pool to allocate, we stop. Max pieces are hit across all events. (fig 3)

(f) Select top record and allocate that customer to that Campaign, before allocating, test if campaign has met budgetary constraint (total pieces or cost budget, you can see they are interchangeable) This is done by maintaining counters for each, which are incremented after a successful allocation, also test to see if he's not exceeded this individual's budget. Maintain a counter per individual for their budget. If either is true, then simply discard the record and past to the next, by moving the arrow down one entry in fig 2.

(g) Delete that record from the global list.

(h) Locate that customer in the remaining records and apply saturation to all other campaign records that he has remaining unallocated. This achieved by using a set of indexes to the original data (Fig 1), so although for illustrative purposes Fig 2 has been re-arranged in the actual implementation we would have a vector of indexes. These are sorted and shuffled during the process. See Fig 4. Let's do the first completely. Sort index says take the fourth record as globally the best expected return. The fourth record is Campaign 4 for customer 123456. We set the campaign id to negative, to show it's been used. In Fig 4. We decrement the global count for campaign 4, and we deduct 1.03 from the global budget for customer (See fig 3) Now we apply saturation, which degrades the expected gain for campaigns 1-3 and 5-7. As we know this is the fourth, we step back to the first and iterate over each campaigns expected gain from 1-7 excluding any negative campaign id's. We have now applied the saturative effect of 4 vs 1, 4 vs 2, 4 vs 3, 4 vs 5, etc. Saturative effect is as per Bibelniks Fig 11. and described in detail, we adopt the same approach for calculating this component. Due to this the expected gain will be reduced for this customer

for all other campaigns proceeding and following the allocated campaign.

Customer Number	Campaign id	Expected Gain	Sort Index
123456	1	0.1	4
123456	2	0.2	9
123456	3	0.4	14
123456	-4	0.9	10
123456	5	0.2	3
123456	6	0.53	7
123456	7	0.52	6
123457	1	0.1	13
123457	2	0.8	11
123457	3	0.63	5
123457	4	0.5	2
123457	5	0.21	12
123457	6	0.52	1
123457	7	0.7	8

Fig 4

(i) Now in a Standard GH you would totally resort the table. We don't. (That's compute intensive, think about it if you have 1m customers over 30 events, that's a 30m record sort, just how it expands...) It turns out that by the nature of saturation, you will suppress all the records remaining by some value, ie: If I give you this campaign it always makes it less likely that things in a near term time horizon will also be given. And we are solving on a say 90 day window, because we can do it that fast.

(j) Ok I've messed my flow a little, but bear with me, I should go to (c) but what I want to do is check the next event is greater than one after the event to be picked, if it isn't I swap the entries, and keep doing that until it is, the technique is similar to that employed in a bubble sort. (Air bubbles rising) Once you know the next is the best you simply continue at (c) Remember this works because you will only make all other events worse, lower value due to the nature of cannibalisation/saturation. Look at Fig 5 this is pretty typical, the application of saturation has adjusted some items lower in the list but have had no effect on the next item to choose. (So traditionally you would sort blindly, in this algorithm you take a lazy evaluation approach) This wastes a significant number of computer cycles for no reason.

Customer Number	Campaign id	Expected Gain	Sort Index (Pass 1)	Sort Index (Pass 2)
123456	1	0.2	4	4
123456	2	0.3	9	9
123456	3	0.565	14	14
123456	4	0.9	10	10
123456	5	0.32	3	6
123456	6	0.53	7	7
123456	7	0.56	6	13
123457	1	0.1	13	11
123457	2	0.8	11	3
123457	3	0.63	5	12
123457	4	0.5	2	2
123457	5	0.21	12	5
123457	6	0.52	1	1
123457	7	0.7	8	8

Fig 5

EXHIBIT B

```
1  //
2  // Vanquish optimiser Started: 4 Dec 2001
3  // Author: David Selby
4  //
5  #include <stdio.h>
6  #include <stdlib.h>
7  #include <stdarg.h>
8  #include <time.h>
9  #include <malloc.h>
10 #include <signal.h>
11 #include <iostream.h>
12 #include <string.h>
13 #include <sys/stat.h>
14 #include "parms.h"
15 #include "fileobj.h"
16 #include "sidefile.h"
17 #include "campaigns.h"
18 #include "saturation.h"
19 #include "xmldoc.h"
20
21 #ifndef AIX
22 #define STDIN_FILENO 0
23 #include <io.h>
24 #include <direct.h>
25 #else
26 #define Sleep sleep
27 #endif
28
29 #ifdef DBVER
30 #include "DBobj.h"
31 #endif
32
33 #ifndef TRUE
34 #define TRUE 1
35 #define FALSE 0
36 #endif
37
```

```

38  #ifdef WIN32
39  typedef unsigned __int64 DWORD64;
40  typedef __int64 INT64;
41  #else
42  typedef unsigned long long DWORD64;
43  typedef long long INT64;
44  #endif
45
46  char *make_tname(char *tempdir);
47  static void sig_end(int signo);
48
49  // Various global parameters set by args to main()
50  #ifndef DOS
51  int in_status = 1000;
52  #else
53  int in_status = 100;
54  #endif
55  int out_status = 0;
56  int hush = TRUE;
57  int max_return = 0;
58  int logfile = 0;
59  int db_connect = TRUE;
60  int status_mark = FALSE;
61  char version[] = {"Version 1.0 \0Compile date here"};
62  char *modis;
63  parms *xmlfile;
64
65  int main(int argc,char **argv)
66  {
67      char      foney[]={'-','?'};
68      int      i,j,rtime,assign_count;
69      char      *cp;
70      struct tm  when;
71      time_t    now,starttime,endtime;
72      char      timestamp[15];
73      int      observations = 10000;
74      double total_spent = 0;

```

```

75 unsigned int total_recount = 0;
76
77 #if AIX
78 if (!isatty(STDOUT_FILENO)) in_status = 0; // If piped don't do status msgs
79 #endif
80
81 #ifdef TIMEBOMB
82 int timeBomb(void);
83
84 if (!timeBomb()) { cout << "ABEND: Module expired." << endl << endl;
85                 cout << "Please contact David Selby" << endl << "SELBY at WINVMB" <<
86 "David_Selby@uk.ibm.com" << endl << "Center for Business Optimization" << endl;
87                 cout << "Hursley Park" << endl << "Winchester" << endl << "Hampshire SO21 2JN" << endl
88 << "England" << endl;
89                 exit(-42);
90             }
91 #endif
92
93 #ifdef MACHONLY
94 int mach(void);
95 if (!mach()) { cout << "ABEND: Machine not licenced to run Vanquish Optimiser" << endl << endl;
96             exit(-42);
97         }
98 #endif
99
100 modis = argv[0];
101 strcat(version, __DATE__);
102
103 // Trap various signals
104 signal(SIGINT, sig_end);
105 signal(SIGTERM, sig_end);
106 #ifdef AIX
107 signal(SIGQUIT, sig_end);
108 signal(SIGIOT, sig_end);
109 signal(SIGBUS, sig_end);
110 signal(SIGSEGV, sig_end);
111 #ifdef SIGDANGER

```

```

112     signal(SIGDANGER,sig_end);
113 #endif
114 #endif
115
116 if (argc == 1) { /* Use an underhanded trick to force the help msg */
117     ++argc; argv[1] = foney;
118 }
119 // Read the command line, Deal with quiet parameter, number of records for internal sort
120 while(--argc > 0) /*&& (*++argv)[0] == '-'*/
121 { ++argv;
122     while(*argv[0] == ' ') (*argv)++;
123     if (*argv[0] == '-')
124     { ++argv[0];
125         if (*argv[0] == '?')
126         { /* Help messages */
127             cout << "Vanquish Optimizer by David Selby" << endl;
128             cout << "Build Date: " << __DATE__ << endl;
129             cout << "Startup Parameters as follows:" << endl;
130             #ifdef DBVER
131             cout << "-noconn Disable connection to Database" << endl;
132             #endif
133             cout << "-quiet Disable all output until first input" << endl;
134             cout << "xxxx where xxxx is a .xml filename" << endl;
135             exit(0);
136         }
137         /* Quiet switch */
138         else if (0 == strcmp(argv[0],"quiet",5))
139         { hush = TRUE;
140             // More parms in this token ?
141             if (strlen(argv[0]) > 5) { (*argv) += 5;
142                 --argv;
143                 ++argc;
144             }
145         }
146         else if (0 == strcmp(argv[0],"noconn",6))
147         { db_connect = FALSE;
148             // More parms in this token ?

```

```

149         if (strlen(argv[0]) > 6) { (*argv) += 6;
150             --argv;
151             ++argc;
152         }
153     }
154     else if (0 == strncmp(argv[0], "status", 6))
155     { status_mark = TRUE;
156       in_status = 1000;
157       // More parms in this token ?
158       if (strlen(argv[0]) > 6) { (*argv) += 6;
159           --argv;
160           ++argc;
161       }
162     }
163     else if (0 == strncmp(argv[0], "instat", 6) && argc)
164     { if (strlen(argv[0]) > 6) { (*argv) += 6;
165         cp = *argv;
166     }
167     else { cp = *++argv;
168         --argc;
169     }
170     i = strtol(cp, (char **)&cp, 10);
171     if (*cp == 'm' || *cp == 'M') i <= 20;
172     if (*cp == 'k' || *cp == 'K') i <= 10;
173     if (i >= 0) in_status = i;
174     // More parms in this token ?
175     if (strlen(cp) > 3) { (*argv) = cp;
176         --argv;
177         ++argc;
178     }
179     }
180     else if (0 == strncmp(argv[0], "outstat", 7) && argc)
181     { if (strlen(argv[0]) > 7) { (*argv) += 7;
182         cp = *argv;
183     }
184     else { cp = *++argv;
185         --argc;

```



```

186         }
187         i = strtol(cp,(char **)&cp,10);
188         if (*cp == 'm' || *cp == 'M') i <= 20;
189         if (*cp == 'k' || *cp == 'K') i <= 10;
190         if (i >= 0) out_status = i;
191         // More parms in this token ?
192         if (strlen(cp) > 3) { (*argv) = cp;
193             --argv;
194             ++argc;
195         }
196     }
197     else if (0 == strcmp(argv[0],"execdir",7) && argc)
198     { if (strlen(argv[0]) > 7) { (*argv) += 7;
199         cp = *argv;
200     }
201     else { cp = *++argv;
202         --argc;
203     }
204     if (chdir(cp))
205         cerr << "unable to change to " << cp << " directory, continuing in current" << endl;
206     else cerr << "changing to " << cp << endl;
207 }
208 // ls -... parm
209 else { while(*argv[0] == ' ') ++argv[0]; /* Elide blanks */
210     xmlfile = new parms(argv[0],&max_return);
211     if (max_return)
212     { if (max_return) printf("Can not find %s",argv[0]);
213         delete xmlfile;
214         return -2;
215     }
216
217 }
218 /* end while args */
219
220 time( &now );
221 when = *localtime( &now );
222

```

```

223     if (!hush)
224     { cout << "Vanquish Optimizer" << endl << "By David Selby" << endl << "IBM Center for Business
225 Optimization" << endl << "Version 1.4" << endl << "Patents pending" << endl;
226         cout << "Build Date: " << __DATE__ << endl;
227         cout << "Start run at " << asctime( &when ) << endl;
228     }
229
230     /* Timestamp is used as a unique run identifier */
231
232     sprintf(timestamp,"%d%02d%02d%02d%02d%02d\0",1900+when.tm_year,1+when.tm_mon,when.tm_md
233 ay,when.tm_hour,when.tm_min,when.tm_sec);
234
235     //
236     // 1) Open campaigns/events file, need the cost per campaign read into core
237     //
238     dataobject *campaign_data = NULL;
239     campaigns *campaign_file = NULL;
240     #if DBVER
241         if (xmlfile->campaign_db_name) campaign_data = new DBobj(xmlfile->campaign_db_name, xmlfile-
242 >campaign_db_userid,xmlfile->campaign_db_passwd,xmlfile->campaign_db_table, xmlfile-
243 >campaign_db_where,NULL,xmlfile->campaign_fields,1,xmlfile->campaign_defn);
244
245     else
246     #endif
247         if (xmlfile->campaign_dataFile) campaign_data = new fileobj(xmlfile->campaign_dataFile,xmlfile-
248 >campaign_fields,1,xmlfile->campaign_defn);
249         if (campaign_data) { if (campaign_data->prepare() == 0)
250             { // Message file failure
251                 // printf(ERR_CAMPAIGN_FILE,xmlfile->campaign_dataFile);
252                 exit(-12);
253             }
254             campaign_file = new campaigns(campaign_data);
255             delete campaign_data;
256         }
257     else { // Message no campaign data
258         exit(-12);
259     }

```

```

260
261     dataobject *customer_data = NULL;
262     sidefile *sidefile_cntrl = NULL;
263     int     ret;
264     // 2) Generate sidefile from customer file, Calculate costs etc. Note zeros and negs
265     // Do initial bucket pass first time to speed up
266     //
267     // Important!!! The code assumes the customer file contains the data pre sorted by customer
268     number and then within event id.
269     // Event ids must be assigned in chronological order. Larger the event id, larger the drop
270     date of the event
271     //
272     #if DBVER
273         if (xmlfile->datafile_db_name) customer_data = new DBobj(xmlfile->datafile_db_name, xmlfile-
274         >datafile_db_userid,xmlfile->datafile_db_passwd,xmlfile->datafile_db_table, xmlfile-
275         >datafile_db_where,xmlfile->datafile_db_order,xmlfile->datafile_fields,1,xmlfile->datafile_defn);
276     else
277     #endif
278         if (xmlfile->datafile_dataFile) customer_data = new fileobj(xmlfile->datafile_dataFile,xmlfile-
279         >datafile_fields,1,xmlfile->datafile_defn);
280
281     if (customer_data) { if (customer_data->prepare() == 0)
282         { // Message file failure
283             exit(-14);
284         }
285         try { sidefile_cntrl = new sidefile(xmlfile->workpath,&ret,customer_data,campaign_file);
286         }
287         catch(int ret)
288             { cerr << "Internal Sidefile creation failed return code is " << ret << endl;
289             exit(-144);
290         }
291         delete customer_data;
292     }
293     else { // Message no customer data
294         exit(-14);
295     }
296

```

```

297 // 3) Generate the sort pointer vector (paged) Burn temporary array (3 passes)
298 // 4) Read the saturation matrix into core.
299 dataobject *sat_data = NULL;
300 saturation *sat_file = NULL;
301 #if DBVER
302     if (xmlfile->sat_db_name) sat_data = new DBobj(xmlfile->sat_db_name, xmlfile->sat_db_userid,xmlfile->
303 >sat_db_passwd,xmlfile->sat_db_table, xmlfile->sat_db_where,NULL,xmlfile->sat_fields,1,xmlfile->
304 >sat_defn);
305     else
306 #endif
307     if (xmlfile->sat_dataFile) sat_data = new fileobj(xmlfile->sat_dataFile,xmlfile->sat_fields,1,xmlfile->
308 >sat_defn);
309
310     if (sat_data) { if (sat_data->prepare() == 0)
311         { // Message file to open saturation file
312             exit(-15);
313         }
314         sat_file = new saturation(sat_data,campaign_file->number,campaign_file->camp_nos);
315         delete sat_data;
316     }
317     else { // Message no saturation data
318         exit(-15);
319     }
320
321     //
322     // Need to handle overrides late otherwise the customer data gets messed up.
323     // (You end up zapping the list of numbers, before they used to make the indexes)
324     campaign_file->do_overrides(xmlfile->campaign_override);
325
326     xmlfile->changestate(1); // Project in pending state
327     // 5) Start full process
328     // sat_file->dump(); // Dump saturation matrix for debugging purposes
329     // sidefile_cntrl->debug(); // Dump sidefile for debugging purposes.
330     // FILE *fgraph = fopen("fgraph.csv","w+"); // Debug
331     observations = sidefile_cntrl->no_records() / 100; // 100 bins
332     if (!hush) cout << "Commencing optimization process" << endl;
333     int  obs = 0;

```

```

334     float spent_graph[100],gain_y[100];
335     gain_y[0] = -1;
336
337     INT64 balance,total,last_balance,balance_pre_min,balance_post_min;
338     DWORD64 gain = 0,gain_net = 0,last_gain = 0;
339
340     if (xmlfile->budget_type == 'n') balance = (INT64)(campaign_file->min_budget +
341 (DWORD64)((double)(INT64)campaign_file->min_budget * ((double)(INT64)xmlfile-
342 >budget/(double)100))); // min +/-x%
343     else if (xmlfile->budget_type == 'x') balance = (INT64)(campaign_file->max_budget -
344 (DWORD64)((double)(INT64)campaign_file->max_budget * ((double)(INT64)xmlfile-
345 >budget/(double)100))); // max +/-x%
346     else balance = (INT64)(100 * xmlfile->budget);
347
348     balance_pre_min = campaign_file->min_budget; // Spend is allocated from two discrete buckets to
349 handle minima constraint
350     balance_post_min = balance - balance_pre_min;
351     starttime = time(NULL);
352
353     total = balance;
354     dataobject *out_data = NULL;
355     if (balance >= (INT64)campaign_file->min_budget)
356     { while(balance > 0 && sidefile_cntrl->getnextval()) // Fetch next best thing to do
357         { int campno = sidefile_cntrl->catno();
358           int campnox = campno-1;
359           unsigned int cost = campaign_file->costs[campnox];
360           unsigned int fcost = 0;
361           // Handle excluded campaigns
362           if (campaign_file->camp_nos[campnox] == -1) campno = -1;
363           // Handle any fixed costs, first allocation forces one time cost
364           if (campno > 0 && campaign_file->pieces[campnox] == 0) fcost = campaign_file-
365 >fcosts[campnox];
366           //
367           // Handle minimum limit constraint, check which money bucket and ensure has funds to
368 support allocation
369           // Otherwise forfeit the allocation
370           //

```

```

371         if (campno > 0 && (!((campaign_file->min_limit[campnox] && (balance_post_min-cost) >= 0) ||
372             (campaign_file->min_limit[campnox] == '0' && (balance_pre_min-cost+fcost) >=
373             0))))
374             campno = -1;
375         //debug: */
376         fprintf(stderr,"campno=%d,balance=%f,satvalue=%f,count=%d,next_choose=%d,current=%d\n",campno,
377             balance,sidefile_cntrl->get_sat_value(),sidefile_cntrl->Count,sidefile_cntrl->next_choose,sidefile_cntrl-
378             >current);
379         // Start with a sanity check to ensure not selecting a selected campaign
380         // Does this exceed the max pieces for this campaign?
381         if (campno > 0 && (campaign_file->max_pieces[campnox] > campaign_file->pieces[campnox]))
382             { //
383                 // Now handle costs
384                 //
385                 // If this is the first allocation then get any fix costs data
386                 // Check to see if this will exceed the global spend
387                 // if yes then pass and try for something cheaper
388                 unsigned int tcost = cost;
389                 if (campaign_file->pieces[campnox] == 0) tcost = fcost+ cost; // First iteration
390             requires inclusion of fixed cost
391
392             if ((balance-tcost) >= 0 &&
393                 // Has the individual reached max budget?
394                 (sidefile_cntrl->getcusbudget() == -1 || // -1 = no individual budget
395                 (sidefile_cntrl->getcusbudget()-cost) > 0)) // Don't count fixed costs against the individual
396                 { // Allocate campaign to individual
397                     DWORD64 g;
398                     sidefile_cntrl->allocated_catno(); // Set as allocated
399                     balance -= tcost; // Reduce the global balance for this event, incl fixed
400             see above
401                     g = (DWORD64)floor(0.5 + 100 * sidefile_cntrl->get_sat_value()); //
402             Incorporate gain (Move decimal pt & round)
403                     gain += g;
404                     g -= tcost;
405                     gain_net += (DWORD64)g; // less costs incl fixed cost
406                     campaign_file->gain[campnox] += (unsigned int)g; // event gain is also incremented
407                     ++campaign_file->pieces[campnox]; // Increase number used

```

```

408         if (!campaign_file->min_limit[campnox]) // Have we reached the min constraint?
409             { balance_pre_min -= cost + fcost;
410               if (campaign_file->min_pieces[campnox] == campaign_file->pieces[campnox])
411                   campaign_file->min_limit[campnox] = '1';
412             }
413         else balance_post_min -= cost;
414
415         // If running with individual budget then reduce also
416         if (sidefile_cntrl->getcusbudget() != -1) sidefile_cntrl->set_custbudget((float)cost/100);
417
418         //
419         // Now saturate across all campaigns
420         // sidefile has not changed order so just need to back up
421         sidefile_cntrl->select_first();
422         assign_count = 0;
423         for(int i = 0; i < campaign_file->number; i++)
424             { if (sidefile_cntrl->catno() > 0 && sidefile_cntrl->get_sat_value() > 0) // If its already
425               negative then don't waste mips
426                   { float unsat = sidefile_cntrl->get_unsat_value() /
427                     ((float)campaign_file->costs[i]/100); // Correct to gains ratio
428                     // sidefile_cntrl->set_sat_value(sidefile_cntrl-
429                     >get_sat_value() - ((float)xmlfile->sat_lever * (sidefile_cntrl->get_unsat_value() * sat_file-
430                     >satmat[i][campnox] + sat_file->satmat[campnox][i] * sidefile_cntrl->get_unsat_value())));
431                     sidefile_cntrl->set_sat_value(sidefile_cntrl->get_sat_value()
432                     - ((float)xmlfile->sat_lever * (unsat * sat_file->satmat[i][campnox] + sat_file->satmat[campnox][i] * unsat)));
433                 }
434                 if (sidefile_cntrl->catno() < 0) ++assign_count;
435                 if (sidefile_cntrl->getnext_srt()) break;
436             }
437         // If we have hit the limit of events for this customer then saturate all
438         remaining available events
439         // Just force it out with a large loss
440         if (xmlfile->maxoffersper && xmlfile->maxoffersper == assign_count)
441             { sidefile_cntrl->select_first();
442               for(int i = 0; i < campaign_file->number; i++) if (sidefile_cntrl->catno() >
443               0 && sidefile_cntrl->get_sat_value() > 0) sidefile_cntrl->set_sat_value((float)-999999999);
444             }

```

```

445     }
446 }
447 sidefile_cntrl->incnextval(); // Move to next entry
448 // Output a diagnostic to help understand progress
449 if (in_status && ((sidefile_cntrl->getnextone() % in_status) == 0))
450 { /*char msg[50];
451     sprintf(msg,"Reading record %ld      ",sidefile_cntrl->getnextone());
452     cout << msg;
453     int i = strlen(msg);
454     while(i-->0) cout << "\b";
455     cout << flush;
456     */
457     endtime = time(NULL);
458     rtime = (int)(double)ceil(((double)(sidefile_cntrl->no_records()-sidefile_cntrl-
459 >getnextone()) * (endtime-starttime)/((sidefile_cntrl->getnextone() == 0)?1:sidefile_cntrl->getnextone()));
460     tm *t = gmtime((time_t *)&rtime);
461     int ts = t->tm_sec;
462     if (t->tm_hour == 0 && t->tm_min == 0 && ts == 0) ts = 1;
463
464     cout << "#" << sidefile_cntrl->getnextone() << ", " << sidefile_cntrl->no_records()
465 << ", " << t->tm_hour << ":" << t->tm_min << ":" << ts << endl << flush;
466     cout.flush();
467     cout.flush();
468     Sleep(1);
469 }
470 // Generate a file for the gains graph
471 if (0 == --observations)
472 { observations = sidefile_cntrl->no_records() / 100; // 100 bins
473     if (last_gain != 0 && last_gain != gain_net)
474     { float gr = (float)((double)(INT64)(gain_net-last_gain)/((double)(INT64)((total-balance)-
475 last_balance));
476         spent_graph[obs] = (float)(total-balance);
477         gain_y[obs] = gr;
478         gain_y[++obs] = -1; // Use -1 as fence to mark end
479         // fprintf(fgraph,"%f,%f\n",total-balance,gr);
480     }
481     last_balance = total-balance;

```



```

482         last_gain = gain_net;
483     }
484 }
485     // Mop up residuals
486     if (last_gain != gain_net && (observations != (int)(sidefile_cntrl->no_records() / 100)))
487     { float gr = (float)((double)(INT64)(gain_net-last_gain)/(double)(INT64)((total-balance)-
488 last_balance));
489         spent_graph[obs] = (float)(INT64)(total-balance);
490         gain_y[obs] = gr;
491         gain_y[++obs] = -1;
492         //fprintf(fgraph,"%f,%f\n",total-balance,gr);
493     }
494     //fclose(fgraph);
495     //
496     // Output statistics from the run
497     //
498     cout << flush;
499     cout << "#" << sidefile_cntrl->no_records() << "/" << sidefile_cntrl->no_records() << ",0:0:0" <<
500 endl << flush << flush;
501
502     if (xmlfile->xmlstats)
503     { cout << "Optimization processing completed" << endl;
504       cout << "<<< Global Statistics from run >>>" << endl;
505       int i;
506       for(i = 0; i < campaign_file->number; i++)
507       { if (campaign_file->camp_nos[i] > 0)
508         { if (campaign_file->pieces[i] < campaign_file->min_pieces[i]) cout << "***";
509           cout << "Campaign " << campaign_file->camp_nos[i] << " Allocated " << campaign_file-
510 >pieces[i] << " min=" << campaign_file->min_pieces[i] << " max=" << campaign_file->max_pieces[i] << "
511 pieces" << endl;
512           total_spent += campaign_file->pieces[i] * campaign_file->costs[i];
513           total_recout += campaign_file->pieces[i];
514         }
515       }
516       cout << "Overall cost incurred " << ((double)(INT64)total_spent/100) << " against a budget of " <<
517 ((double)(INT64)total/100) << " balance is " << ((double)(INT64)balance/100) << endl;

```

```

518         cout << "Estimated gross gain is " << ((double)(INT64)gain/100) << " Net gain is " <<
519         ((double)(INT64)gain_net/100) << endl;
520         cout << "Estimated profit margin is " << 100 * ((double)(INT64)gain_net/((double)(INT64)gain) <<
521         "%" << endl;
522
523
524         xmldoc *statsdoc = new xmldoc("stats"); // Create document
525         if (statsdoc)
526         { char numbuf[60];
527           sprintf(numbuf, "%.2f", ((float)(INT64)total)/100);
528           statsdoc->clog_off = TRUE;
529           statsdoc->chg("\\stats\\description", xmlfile->description);
530           statsdoc->add("\\stats\\runid", timestamp);
531           statsdoc->add("\\stats\\projectid", xmlfile->projectid);
532           statsdoc->add("\\stats\\budget", numbuf);
533           sprintf(numbuf, "%.2f", ((float)(INT64)balance)/100);
534           statsdoc->add("\\stats\\balance", numbuf);
535
536           // Store Gains graph
537           i = 0;
538           char *buf = new char[sizeof("\\stats\\gaingraph@pnt=1000");
539           do { sprintf(numbuf, "%.2f, %.2f", spent_graph[i]/100.0, gain_y[i]);
540               sprintf(buf, "\\stats\\gaingraph@pnt=%d", i+1);
541               statsdoc->add(buf, numbuf);
542           } while(gain_y[++i] != -1);
543
544           sprintf(buf, "\\stats\\graph@pnts=%d", i+1);
545           statsdoc->add(buf, NULL);
546
547           delete [] buf;
548
549           for(i = 0, j = 1; i < campaign_file->number; i++)
550           { if (campaign_file->camp_nos[i] > 0)
551             { char ev[sizeof("\\stats\\event@id=9999\\description")+3];
552               sprintf(numbuf, "%d", j); //DAS -- Caryn can't take zero origin output
553               statsdoc->add("\\stats\\event@id", numbuf);
554               sprintf(ev, "\\stats\\event@id=%d\\eventid", j);

```

```

555         sprintf(numbuf,"%d",campaign_file->camp_nos[i]);
556         statsdoc->add(ev,numbuf);
557         sprintf(ev,"\\stats\\event@id=%d\\description",j);
558         statsdoc->add(ev,campaign_file->desc[i]);
559         sprintf(ev,"\\stats\\event@id=%d\\minpieces",j);
560         sprintf(numbuf,"%d",campaign_file->min_pieces[i]);
561         statsdoc->add(ev,numbuf);
562         sprintf(ev,"\\stats\\event@id=%d\\maxpieces",j);
563         sprintf(numbuf,"%d",campaign_file->max_pieces[i]);
564         statsdoc->add(ev,numbuf);
565         sprintf(ev,"\\stats\\event@id=%d\\allocated",j);
566         sprintf(numbuf,"%d",campaign_file->pieces[i]);
567         statsdoc->add(ev,numbuf);
568         sprintf(ev,"\\stats\\event@id=%d\\fcost",j);
569         sprintf(numbuf,"%0.2f", (float)campaign_file->fcosts[i]/100);
570         statsdoc->add(ev,numbuf);
571         sprintf(ev,"\\stats\\event@id=%d\\cost",j);
572         sprintf(numbuf,"%0.2f", (float)campaign_file->costs[i]/100);
573         statsdoc->add(ev,numbuf);
574         sprintf(ev,"\\stats\\event@id=%d\\gain",j++);
575         sprintf(numbuf,"%0.2f", (float)campaign_file->gain[i]/100);
576         statsdoc->add(ev,numbuf);
577     }
578 }
579 sprintf(numbuf,"%d",j-1);
580 statsdoc->add("\\stats\\events@no",numbuf);
581 char *s = new char[sizeof("stats.xml")+1];
582 strcpy(s,"stats.xml");
583 statsdoc->setfilename(s);
584 statsdoc->flush();
585 statsdoc->clog_off = FALSE;
586 delete statsdoc;
587 }
588 }
589     else { sidefile_cntrl->restart();
590         int total_contacts = 0;
591         while(sidefile_cntrl->next_custno()) // for each customer input

```

```

592         { int contact = 0;
593             do { if (sidefile_cntrl->catno() < 0) contact = 1; } while(!sidefile_cntrl-
594 >getnext_srt());
595             total_contacts += contact;
596         }
597         cout << "##{"budget" : \" << ((float)(INT64)total/100.0) << "\",\"balance\" : \" <<
598 ((float)(INT64)balance/100) << "\",\"obs\" : \" << obs << "\",\"events\" : \" << campaign_file->number <<
599 "\",\"contacts\" : \" << total_contacts << "\",\"customers\" : \" << sidefile_cntrl->total_customers << "\"}";
600         for(i = 0; i < obs; i++) cout << "\",\"point\" : \" << spent_graph[i]/100.0 << "," <<
601 gain_y[i] << "\"}";
602         for(i = 0; i < campaign_file->number; i++)
603             if (campaign_file->camp_nos[i] > 0)
604                 { cout << "\",\"event\" : \" << campaign_file->camp_nos[i] << "\",\"desc\" : \" <<
605 campaign_file->desc[i] << "\",\"mediacode\" : \" << campaign_file->mediacode[i] << "\";
606                 cout << "\",\"minpieces\" : \" << campaign_file->min_pieces[i] << "\",\"maxpieces\" : \" <<
607 campaign_file->max_pieces[i] << "\",\"allocated\" : \" << campaign_file->pieces[i] << "\",\"fcost\" : \" <<
608 campaign_file->fcosts[i] << "\",\"cost\" : \" << campaign_file->costs[i] << "\",\"gain\" : \" << campaign_file-
609 >gain[i] << "\"}";
610                 total_spent += campaign_file->pieces[i] * campaign_file->costs[i];
611                 total_recout += campaign_file->pieces[i];
612             }
613         cout << endl;
614         //      cout << "\n" << endl;
615     }
616     #if DBVER
617         // Write stats to database if required
618         if (xmlfile->runstats_db_name)
619             { // Additionally insert into a couple of database tables
620                 out_data = new DBObj(xmlfile->runstats_db_name,xmlfile->runstats_db_userid, xmlfile-
621 >runstats_db_passwd,xmlfile->runstats_db_table1, NULL,NULL,xmlfile->runstats_fields1,2,xmlfile-
622 >runstats_defn);
623                 if (out_data) { out_data->setbuffer(1);
624                     if (out_data->prepare())
625                         { out_data->setrecs(1);
626                             field *f = xmlfile->runstats_fields1;
627                             out_data->setString("runid",timestamp);
628                             if (xmlfile->description) out_data->setString("description",xmlfile->description);

```

```

629         out_data->setString("projectid",xmlfile->projectid);
630         out_data->setFloat("budget",(float)(INT64)total/100);
631         out_data->setFloat("balance",(float)(INT64)balance/100);
632                 out_data->setInt("points",obs);
633                 out_data->setVect("gaingraphx",spent_graph);
634                 out_data->setVect("gaingraphy",gain_y);
635         out_data->setInt("events",campaign_file->number);
636         out_data->writenext();
637         delete out_data;
638         out_data = new DBObj(xmlfile->runstats_db_name,xmlfile->runstats_db_userid,
639 xmlfile->runstats_db_passwd,xmlfile->runstats_db_table2,NULL,NULL,xmlfile->runstats_fields2,2,xmlfile-
640 >runstats_defn);
641         if (out_data) { out_data->setbuffer(j);
642                 if (out_data->prepare())
643                 { out_data->setrecs(j);
644                         field *f = xmlfile->runstats_fields2;
645                         for(i = 0; i < campaign_file->number; i++)
646                         { if (campaign_file->camp_nos[i] > 0) // Handle overrides
647                         { out_data->setString("runid",timestamp);
648                                 out_data->setInt("eventid",campaign_file->camp_nos[i]);
649                                 out_data->setString("description",campaign_file->desc[i]);
650                                 out_data->setInt("minpiece",campaign_file->min_pieces[i]);
651                                 out_data->setInt("maxpiece",campaign_file->max_pieces[i]);
652                                 out_data->setInt("alloc",campaign_file->pieces[i]);
653                                 out_data->setFloat("cost",(float)campaign_file->costs[i]/100);
654                                 out_data->setFloat("fcost",(float)campaign_file->fcosts[i]/100);
655                                 out_data->setFloat("gain",(float)campaign_file->gain[i]/100);
656                                 out_data->writenext();
657                         }
658                 }
659         }
660         delete out_data;
661     }
662 }
663 }
664
665 }

```

```

666 #endif
667     // 6) Generate output file
668     #if DBVER
669         if (xmlfile->output_db_name) out_data = new DBobj(xmlfile->output_db_name, xmlfile-
670 >output_db_userid, xmlfile->output_db_passwd,xmlfile->output_db_table, xmlfile-
671 >output_db_where,NULL,xmlfile->output_fields,2,xmlfile->output_defn);
672     else
673     #endif
674         if (xmlfile->output_dataFile) out_data = new fileobj(xmlfile->output_dataFile,xmlfile-
675 >output_fields,0,xmlfile->output_defn);
676
677     if (out_data) { // Simply run thru the sidefile from top to bottom and output selected campaigns
678         if (out_data->prepare())
679             { // Return can happen because of failure or Suppress, complete run anyway.
680                 out_data->setrecs(total_recout);
681                 sidefile_cntrl->restart();
682                 field *f = xmlfile->output_fields;
683                 int bvect = FALSE;
684                 // Check to see if this is the boolean output style
685                 while(f) { if (f->type == 'b' && f->repeat)
686                     { bvect = TRUE;
687                       f->val.str = new char[f->repeat*2];
688                       memset(f->val.str,' ',f->repeat*2);
689                     }
690                     f = f->next;
691                 }
692
693                 while(sidefile_cntrl->next_custno()) // for each customer input
694                     { //for(int i = 0; i < campaign_file->number; i++)
695                         // { if (bvect) // This was the original debug method, output a series of 0 and 1's
696                             // { out_data->setBoolean("rvect",i,sidefile_cntrl->catno_inc() < 0);
697                             //     if (i == (campaign_file->number-1)) out_data->writenext();
698                             // }
699                             // else
700                                 do { if (sidefile_cntrl->catno() < 0)
701                                     { // out_data->setString("projectid",xmlfile->projectid);
702                                         // out_data->setString("rundate",timestamp);

```

```

703         out_data->setString("custid",sidefile_cntrl->get_custno());
704         out_data->setInt("eventid",campaign_file->camp_nos[(-sidefile_cntrl-
705 >catno()-1)];
706         if (out_data->writenext()) break; // Drop out if failure
707     }
708         } while(!sidefile_cntrl->getnext_srt());
709     }
710     // if (bvect) while(f) { if (f->type == 'b' && f->repeat) delete [] f->val.str;
711     //         f = f->next;
712     //     }
713 }
714 }
715 }
716 else cout << "Error: Minimum cost of campaigns exceeds imposed global budget" << endl;
717
718 xmlfile->add("\\project\\runid",timestamp); // Put the run id into the parms file
719 xmlfile->changestate(2,modis,version,(long)diffime(time(NULL),now)); // Project in completed state
720 (Useful to know runtime for reference)
721
722 delete campaign_file;
723 delete sidefile_cntrl;
724 delete sat_file;
725 delete out_data;
726
727 if (!hush)
728     { time( &now );
729       when = *localtime( &now );
730       cout << "Finish run at " << asctime( &when ) << endl;
731     }
732
733 return 0;
734 }
735
736 static void sig_end(int signo)
737 {
738     struct tm      when;
739     time_t         now;

```

```

740
741     switch(signo) {
742     case SIGINT: cerr << "Interrupt Signal received" << endl; break;
743     case SIGTERM: cerr << "Terminate Signal received" << endl; break;
744     #ifdef AIX
745     case SIGQUIT: cerr << "Quit Signal received" << endl; break;
746     case SIGIOT: cerr << "I/O Terminate Signal received" << endl; break;
747     case SIGBUS: cerr << "Bus error Signal received" << endl; break;
748     case SIGSEGV: cerr << "Segmentation fault Signal received" << endl; break;
749     #ifdef SIGDANGER
750     case SIGDANGER: cerr << "Danger Signal received, machine low on resource" << endl; break;
751     #endif
752     #endif
753     default: cerr << "Unknown signal caught no=%d" << signo << endl;
754     }
755
756     cout << endl << "ERROR: Forced run abort" << endl;
757     time( &now );
758     when = *localtime( &now );
759     cout << "Finish run at " << asctime( &when ) << endl;
760
761     delete xmlfile;
762     exit(-16708);
763 }
764
765 char *make_tname(char *tempdir)
766 {
767     struct stat stbuf;
768     char temp_name[] = {"VQtmp000"};
769     char *tempfile = temp_name;
770     int name_cnt = 1, handle = -1, append;
771
772     // Sort out temporary directory name
773     if (tempdir) { // Do we need to append a / or \ ?
774         append = strlen(tempdir);
775         if (tempdir[append-1] == '/' || tempdir[append-1] == '\\') append = 1;
776         else append = 2;

```



```

777     }
778     tempfile = (tempdir) ? new char[append+strlen(tempdir)+strlen(temp_name)] :
779         new char[1+strlen(temp_name)];
780     // Locate a free slot
781     while(handle == -1 && name_cnt < 999)
782     { if (tempdir) { strcpy(tempfile,tempdir);
783         #ifdef AIX
784         if (append == 2) strcat(tempfile,"/");
785         #else
786         if (append == 2) strcat(tempfile,"\\");
787         #endif
788         strcat(tempfile,temp_name);
789     }
790     else strcpy(tempfile,temp_name);
791     handle = stat(tempfile, &stbuf);
792     if (handle == -1) break;
793     else { if (temp_name[7] == '9')
794         { temp_name[7] = '0';
795         if (temp_name[6] == '9')
796         { temp_name[6] = '0';
797         if (temp_name[5] == '9') temp_name[5] = '0';
798         else ++temp_name[5];
799         }
800         else ++temp_name[6];
801         }
802         else ++temp_name[7];
803     }
804     ++name_cnt; handle = -1;
805 }
806
807 if (name_cnt == 999) { cerr << "Error: Cannot find free temporary name\nPlease delete FStmpnnn files"
808 << endl;
809     return NULL;
810 }
811 return tempfile;
812 }
813

```

```

814  #ifdef TIMEBOMB
815  int timeBomb(void)
816  {
817      time_t    endtime ;      /* time of expiry          */
818      struct tm  endtm ;       /* time of expiry structure */
819      endtm.tm_sec = 0;
820      endtm.tm_min = 0;
821      endtm.tm_hour = 0;
822      endtm.tm_mday = 01;      /* date at which it first becomes invalid*/
823      endtm.tm_mon = 10;       /* note - zero origin, Jan = 0    !!! */
824      endtm.tm_year = 99;      /* about six months hence          */
825      endtm.tm_wday = 0;
826      endtm.tm_yday = 0;
827      endtm.tm_isdst = 0;
828      return (int)(time(NULL) < mktime(&endtm));
829  } /* end timeBomb */
830  #endif
831
832  #ifdef MACHONLY
833  int mach(void)
834  {
835      FILE      *machchk;
836      char      c;
837
838      machchk = popen("hostname | xargs host | grep -c ' 10.'", "r");
839      fread(&c, 1, 1, machchk); /* get results of command */
840      pclose(machchk);         /* Close pipe */
841
842      return (int) (c != '0'); /* 1 =OK 0= non licenced network */
843  }
844  #endif
845
846  char *timemod(char *time)
847  {
848      static char buf[30];
849
850      strcpy(buf,time);

```

```
851     char *t = strchr(buf, '\n');
852     if (t) *t = '\0';
853
854     return buf;
855 }
856
857
```